Multi-Robot Learning Dynamic Obstacle Avoidance in Formation With Information-Directed Exploration

Junjie Cao, Yujie Wang, Yong Liu, *Member, IEEE*, and Xuesong Ni

***Abstract*—This paper presents an algorithm that generates distributed collision-free velocities for multi-robot while maintain formation as much as possible. The adaptive formation problem is cast as a sequential decision-making problem, which is solved using reinforcement learning that trains several distributed policies to avoid dynamic obstacles on the top of consensus velocities. We construct the policy with Bayesian Linear Regression based on a neural network (called BNL) to compute the state-action value uncertainty efficiently for sequential decision making. The information-directed sampling is applied in our BNL policy to achieve efficient exploration. By further combining the distributional reinforcement learning, we can estimate the intrinsic uncertainty of the state-action value globally and more accurately. For continuous control tasks, efficient exploration can be achieved by optimizing a policy with the sampled action value function from a BNL model. Through our experiments in some contextual Bandit and sequential decision-making tasks, we show that exploration with the BNL model has improved efficiency in both computation and training samples. By augmenting the consensus velocities with our BNL policy, experiments on Multi-Robot navigation demonstrate that adaptive formation is achieved.**

***Index Terms*—Multi-Robot formation, dynamic obstacle avoidance, reinforcement learning, exploration.**

I. Introduction

THE growing popularity of multiple robots in tasks, such as search and rescue, fire-fighting, reconnaissance, and surveillance, has been driving research towards their autonomous navigation. Multiple robots are usually designed to achieve tasks by keeping formation, where coordination and collision avoidance is needed. Many works in robotics have addressed the problem of collision avoidance in robot navigation with moving obstacles [1]–[5]. While there has been impressive progress in the past decade, coordinated autonomous navigation remains challenging, particularly in uncertain, dynamic environments cohabited by dynamic obstacles. The challenges arise because the decentralization of multiple robots is needed for flexibility and the dynamic obstacles' intents are typically not known. Based on optimal control theory, Sequential Convex Programming (SCP) [6] has been used for multi-robot path planning with good optimality guarantees theoretically. However, the computational complexity limits the applications of such centralized optimization-based approaches to few static obstacles and does not scale well with a large number of robots [7]. It is difficult to program a distributed strategy to prevent each robot colliding and keep in coordination with others. Instead, recent approaches have used deep learning to model the complex interactions among multiple robots and obstacles, based on which the motion controller is learned with samples [8]–[12]. Moreover, the strategy approximated with neural networks can generalize to different scenarios without tedious parameter tuning, especially for multiple robots in formation where not only the collision avoidance but also the consensus should be considered. Reinforcement learning (RL), a popular machine learning method, can be used to construct such collision avoidance policy by trial-and-error without labeled data. The sample efficiency in robot learning process is the focus of RL approaches.

In this paper, the navigation of multiple robots is performed with consensus-based formation controller and the collision avoidance is modeled as a sequential decision-making problem which is solved by training a neural network policy with reinforcement learning. Recent developments in deep learning have sparked renewed interest in sequential decision making, providing neural networks for value function approximation. Neural networks have proven powerful and flexible in mapping directly from complex states to the value estimate of actions. However, it remains difficult to quantify its uncertainty on new data, which is essential for efficient exploration in sequential decision making.

There are a lot of works in Bayesian deep learning that dedicate to quantify neural network uncertainty. As a typical example, the Bayesian Neural Network maintains a Gaussian distribution over the parameters of the neural network [13], while it remains difficult in training both the mean and variance of all parameters. One ongoing line of research within Bayesian deep learning aims to develop more practical and approximate Bayesian inference schemes for neural networks, such as Monte-Carlo Dropout [14] and Ensemble [15], [16]. Though these approximate methods are easy to implement, their uncertainty estimations rely on the sampling from the approximated posterior. It is assumed each sampled individual neural network will converge to output similarly where input data has been observed, but predictions will be

Manuscript received July 29, 2021; revised October 7, 2021; accepted November 4, 2021. This work was supported by National Natural Science Foundation of China (NSFC) under Grant 62088101 Autonomous Intelligent Unmanned Systems. *(Correspondence author: Yong Liu.)*

Junjie Cao and Yong Liu are with the Institute of Cyber Systems and Control, Zhejiang University, Hangzhou 310058, China (e-mail: cjunjie@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Yujie Wang is with the Interdisciplinary Center for Social Sciences (ICSS), Zhejiang University, Hangzhou 310058, China (e-mail: 11801020@zju.edu.cn).

Xuesong Ni is with the Beijing Electro-mechanical Engineering Institute, Beijing 100074, China (e-mail: 345474136@qq.com).

Digital Object Identifier 10.1109/TETCI.2021.3127925

diverse elsewhere. Thus, the variance of these sampled neural networks' outputs can be interpreted as uncertainty. However, these uncertainty estimation methods suffer from the complexity in model training process. The Bayesian liner regression model has analytic expressions for uncertainty calculation without any posterior approximation or sampling and is studied in tabular reinforcement learning with proven bounds on the expected regret [17]–[19]. Though with efficient uncertainty calculation, the Bayesian linear regression model still lacks representational power. We propose to combine both efficient uncertainty calculation and powerful representation by performing the Bayesian linear regression on the top of the output of a neural network.

Balancing exploration and exploitation is a fundamental challenge of achieving sample efficiency in sequential decision-making. Having an understanding of what is not yet known or well understood is critical to efficient exploration. So a natural solution to trade-off exploration and exploitation originates from tracking the uncertainty about the approximated model and directing where to explore with the uncertainty [20]–[22]. Beyond tracking the uncertainty, Information Directed Sampling (IDS) [23] also estimates the instantaneous regret. In the Multi-Armed Bandit problem, IDS constructs a regret-information ratio, considering both uncertainty and regret, to balance between minimizing regret and promoting information gain at every decision step.

The key contributions of this paper are three-fold: We proposed a new reinforcement learning algorithm that achieves efficient exploration both in sampling and computation; We introduced a novel distributed collision avoidance strategy for formation of multiple robots and provide an empirical analysis in the simulation of multiple robots point-to-point navigation; To the best of our knowledge, our method is the first that in the distributed formation control field where reinforcement learning is used for dynamic obstacle avoidance. This paper is organized as follows: We review previous related works in Section II and introduce the background in Section III. In order to achieve efficient exploration both in sampling and computation, in Section IV, we propose to approximate the action value with the Bayesian linear regression based on the extracted feature with the neural network. Then in Section V, we extend the IDS strategy to the contextual bandit problem and Markov decision process (MDP) to construct information-directed exploration. By optimizing a policy with a sampled value function, we achieve efficient exploration in the continuous control for adaptive formation in Section VI. Experiments in Contextual Bandit, MDP, Continuous Control and adaptive formation are introduced in Section VII. Finally, we conclude the paper and discuss the future work in Section VIII.

II. Related Work

Combining sampling-based RRT [24] and grid-based forward search A*[25], MARRT* [26] and DMA-RRT [27] are framed as Multi-agent motion planning algorithms. However, these algorithms are heuristic and require additional assumptions, such as sparsity in the environment. From the optimization perspective, Multi-Robot path planning problems have been framed as Mixed Integer Linear Programs (MILPs) [28] and Sequential Convex Programming (SCP) [6]. Such centralized optimization-based approaches often have good optimality guarantees theoretically. Still, their computational complexity limits their applications to few static obstacles and does not scale well with a large number of robots [7]. Optimal Reciprocal Collision Avoidance (ORCA) [29] and its extensions [30] have been popular in multi-robots collision avoidance. However, the search of collision avoidance velocity is complex. Another line of the approach is imitation learning, where robots learn to avoid obstacles from demonstrations of desired behaviors. By imitating the ROS navigation package's demonstrations, a policy was trained to generate motion commands [31]. However, the performance of the imitated policy is seriously constrained by the quality of the demonstration. To overcome such limitation, deep reinforcement learning method is proposed to train a mapless motion planner [32] and achieve multi-robot collision avoidance [8], [9]. This paper also resort to reinforcement learning for Multi-Robot obstacle avoidance problem. Different from previous works, our distributed policies are trained to generate collision-free velocities to coordinate the motion of robots in formation. And we focus on improving the efficiency of robot learning process.

In solving sequential decision-making problem, heuristic exploration strategies that rely on adding noise have achieved impressive performance, such as ϵ-greedy [33] and parameter randomization [34]. These exploration strategies without direction are inefficient. A lot of works use state novelty to direct the exploration and achieve human-level performance in computer games [35]. Compared with state novelty, value function uncertainty may be a more promising guidance of exploration for value maximization. Upper Confidence Bound (UCB) is a prominent uncertainty-directed exploration strategy that originates from bandit problem [21] and also achieves impressive success in reinforcement learning [36]. Thompson Sampling [20] is also a principled exploration strategy directed by uncertainty. It samples actions according to their posterior probability of being optimal and often provides better empirical results than UCB [37]. A log of works [18], [38] extended Thompson sampling to value-based reinforcement learning.

Information Gain has been used for active data selection [39]. Beyond that, Information Directed Sampling (IDS) [23] provides a framework to design efficient exploration strategies that balance the estimated instantaneous regret and the expected information gain. Through the choice of an appropriate information-gain function and approximating the value function with ensemble neural networks, previous work applied IDS in the exploration for deep reinforcement learning [40]. However, uncertainty estimation with multiple neural networks in ensemble results in extensive computation. Bayesian DQN (BDQN) [41] uses Bayesian linear regression on the top of one neural network instead of learning multiple Q-networks, resulting in an approximated posterior over Q-function. The BDQN deploys Thompson Sampling on the approximated posteriors to balance exploration and exploitation. However, sampling the parameter from the high dimensional Gaussian posterior distribution still bear with high computational complexity.

In this paper, we approximate the action value with Bayesian linear regression on the top of one neural network. So we

can achieve an efficient posterior approximation of the value function. To further improve the efficiency of both computation and sample, we define the information gain and regret in the Bayesian model and extend the IDS strategy to information-directed exploration in the contextual bandit and MDP. And we also find that Thompson sampling can be applied in the Bayesian model to achieve the uncertainty-directed exploration in continuous control tasks.

III. Background

A. Consensus-Based Formation Control

A weighted directed graph $G = (\mathcal{V}, \mathcal{E})$ with the node set $\mathcal{V} = \{1, \ldots, n\}$ and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is adopted to describe the communication topology among the n robots. An edge $(i, j) \in \mathcal{E}$ indicates that node j can receive information from node i, and node i is a neighbor of node j. The set of all neighbors of node i is denoted by N_i. A directed path from node i_1 to node i_p is a sequence of ordered edges in the form of $(i_m, i_{m+1}), m = 1, \ldots, p-1$. A directed graph is said to contain a directed spanning tree if there exists at least a node such that the node has directed paths to all other nodes in $\mathcal{G}$. The weighted adjacency matrix $\mathcal{A} = [a_{ij}] \in R^{nn}$ associated with $\mathcal{G}$ is defined by $a_{ij} = 0$ if $(j, i) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. The Laplacian matrix $\mathcal{L} = [l_{ij}] \in R^{nn}$ associated with $\mathcal{G}$ is defined as $l_{ii} = \sum_{j \in \mathcal{N}_i} a_{ij}$ and $l_{ij} = -a_{ij}, i \neq j$.

Here we derive a distributed control law, designed to drive a set of robots with single integrator in formation. Consider a set of N robots modeled by single integrators: $\dot{x}_i(t) = u_i(t), i = 1, \ldots, N$ where $x_i(t) \in \mathbb{R}^n$ and $u_i(t) \in \mathbb{R}^n$ denote the position and velocity, respectively, of the i-th robot at time t. Moreover, let the set of vectors x_{id}, with $i = 1, \ldots, N$, define a desired formation, where the desired displacement between the robot i and the robot j corresponds to $d_{ij} := x_{id} - x_{jd}$. Consider the set of robots communicate according to a communication structure defined by the adjacency matrix $A \in \mathbb{R}^{nn}$. Moreover, let the set of vector x_{id} that define the desired formation be given. If there exist a directed path from v_i to all the other nodes, then for the closed loop system with

$$u_i(t) = \frac{1}{\eta_i} \sum_{j=1}^{N} a_{ij} \left[\dot{x}_j(t) - \gamma \left(x_i(t) - x_j(t) - d_{ij} \right) \right] \tag{1}$$

with $i = 1, \ldots, N$, $\eta_i := \sum_{j=1}^{N} a_{ij}$, $\gamma > 0$, and $d_{ij} := x_{id} - x_{jd}$, the formation is asymptotically satisfied, i.e., $x_i(t) - x_j(t) \to x_{id} - x_{jd}$ as t goes to infinity.

B. Bayesian Linear Regression

We have a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\} = (\mathbf{X}, \mathbf{y})$, where $\mathbf{x}$ denotes an input vector of dimension d, y denotes a scalar output. $\mathbf{x}_i$ and y_i for all n cases are aggregated in the $d \times n$ design matrix $\mathbf{X}$ and the vector $\mathbf{y}_{n \times 1}$ respectively. Function $\phi(\mathbf{x})$ maps a d-dimensional input vector $\mathbf{x}$ into an m dimensional feature space. The matrix $\Phi(\mathbf{X})_{m \times n}$ is the aggregation of columns $\phi(\mathbf{x})$ for all cases in the training set. And the vector of parameters has length m. For clarity, we define $\phi_* = \phi(\mathbf{x}_*), \Phi = \Phi(\mathbf{X})$.

The regression model with Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ can be represented with

$$f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}, \quad \mathbf{y} = f(\mathbf{x}) + \varepsilon. \tag{2}$$

Assume a Gaussian prior of weights: $\mathbf{w} \sim \mathcal{N}(\mu_0, \Sigma_0)$, and calculate the likelihood $p(\mathbf{y}|X, \mathbf{w}) = \mathcal{N}(\phi(\mathbf{x})^\top \mathbf{w}, \sigma^2 I)$. Then the posterior distribution over the weights is Gaussian:

$$\begin{aligned} \mathbf{w}|\mathbf{X}, \mathbf{y} &\sim \mathcal{N}\left(\frac{1}{\sigma^2} \mathbf{A}^{-1} \Phi \mathbf{y} + \mathbf{A}^{-1} \Sigma_0^{-1} \mu_{\mathbf{0}}, \Sigma \right), \\ \mathbf{A} &= \sigma^{-2} \Phi \Phi^T + \Sigma_0^{-1}, \quad \Sigma = \mathbf{A}^{-1}. \end{aligned} \tag{3}$$

By averaging the output of all possible linear models w.r.t. its posterior, the predictive distribution for $f_* \triangleq f(\mathbf{x}_*)$ is

$$f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}\left(\frac{1}{\sigma^2} \phi_*^T \mathbf{A}^{-1} \Phi \mathbf{y} + \phi_*^T \mathbf{A}^{-1} \Sigma_0^{-1} \mu_{\mathbf{0}}, \phi_*^T \Sigma \phi_* \right).$$

C. Bandit Problem and Markov Decision Process

In the Multi-Armed Bandit problem, there are no environmental states. The agent simply learns the value of each action: Q_a and chooses the action a according to all actions' values. Contextual Bandit introduces the concept of state x. The state consists of a description of the environment that the agent can use to make decision. The goal of the agent in the contextual bandit is to learn the best action a^* not just for a single state, but for any number of them and achieve the maximum cumulative reward over time. The expected regret in contextual bandit is defined as $\Delta(a) := \mathbb{E}[Q_{a^*}(x) - Q_a(x)]$, which is the loss in reward for choosing a suboptimal action a in state x. In contextual bandit, states are independent of the previous states or actions, thus the value of one action is evaluated with the instant random reward the agent receives, without the needs to consider the delayed rewards.

In Markov Decision Process (MDP), an agent in state x_t takes action a_t, receives reward $r_t = R(x_t, a_t)$ and gets transitioned to the next state x_{t+1} with probability $p(x_{t+1}|x_t, a_t)$ which depends on the previous state and action. The action value in MDP is defined as the discounted expected cumulative reward that agent will receive by first taking action a in state x and following policy π thereafter:

$$Q^\pi(x, a) = \mathop{\mathbb{E}}_{a_t \sim \pi(\cdot|x_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) | x_0 = x, a_0 = a \right]. \tag{4}$$

D. Modeling Value Function With Distribution

Instead of the expected action value (Eq. (4)), we can estimate a distribution of the action value as a random variable of the discounted cumulative rewards in the future, denoted as: $Z^\pi(x, a) = \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t)$. As a practical algorithm, $Z^\pi(x, a)$ can be assumed to be Gaussian [42] with parameterized mean $Q^\pi(x, a)$ and fixed standard deviation. Without Gaussian assumption, C51 [43] uses a parametric family of distribution

$Z_\theta(x,a)$ to approximate full distribution $Z^\pi(x,a)$, with parameter θ. Define a projection operator Π:

$$\Pi Z := \arg\min_{Z_\theta} D_{KL}(Z, Z_\theta),$$

where $D_{KL}(Z_1, Z_2)$ is the Kullback-Leibler divergence between distribution Z_1 and Z_2. Thus, Π projects a distribution Z into Z_θ in the parametric family with the smallest discrepancy from Z. In C51 the distribution Z is updated as:

$$Z^{(t+1)}(x,a) \leftarrow \Pi \mathcal{H}^* Z^{(t)}(x,a).$$

where $\mathcal{H}^*$ is the distributional Bellman optimality operator for some optimal policy π^*. For a given pair (s_t, a_t), we can first select a greedy action for next state $a' = \arg\max_a \mathbb{E}[Z_\theta(x_{t+1}, a)]$, then update the distribution $Z_\theta(s_t, a_t)$ to match the target distribution:

$$Z = R(x_t, a_t) + \gamma Z_\theta\left(x_{t+1}, a'\right) \tag{5}$$

by minimizing the discrepancy

$$\min_\theta D_{KL}\left(Z, Z_\theta\left(x_t, a_t\right)\right). \tag{6}$$

In practice, we can only sample from the target distribution $x_i \sim R(x_t, a_t) + \gamma Z_\theta(x_{t+1}, a'), 1 \le i \le N$, then estimate the target distribution with the empirical distribution:

$$\hat{Z} = \Sigma_{i=1}^N \frac{1}{N} \delta(x - x_i),$$

where $\delta(x - x_i)$ is the dirac distribution. Then, Eq. (6) reduces to minimizing $D_{KL}(\hat{Z}, Z_\theta)$.

E. Policy Gradient Algorithm

Approximating value function can solve Markov Decision Process where discrete action should be chosen. While continuous control problems are often tackled using policy gradient or actor-critic methods [44], [45], where a separate parameterized "actor" policy π is learned in addition to the value function "critic" Q.

Policy gradient methods maximize the expected cumulative reward by estimating the performance gradient with respect to the policy parameter vector and updating the parameter vector with gradient ascent. In stochastic policy gradient methods [46], stochastic policy samples from a Gaussian distribution $\pi_\alpha \sim N(\mu^\pi(x), \sigma^\pi(x)^2 I)$ with μ^π and σ^π parameterized by α. In deterministic policy gradient methods such as DPG [44] and DDPG [45], the critic estimates the action-value function $Q(x,a)$ using off-policy data which is sampled by a noisy policy. The noisy policy improves the exploration by adding additive action noise to deterministic policy: $\hat{\pi}_\alpha(x) = \pi_\alpha(x) + w$, where $w \sim \mathcal{N}(0, \delta^2 I)$. With the noise variance δ^2 annealing during training process, deterministic policy gradient methods make a trade-off between exploration and exploitation.

IV. Bayesian Neural Linear Policy

We propose to approximate $\phi(\mathbf{x})$ in Eq. (2) with a neural network to project the input $\mathbf{x}$ into the feature space, which is equivalent to performing the Bayesian linear regression on the top of the output of a deep neural network. We call this full model Bayesian Neural Linear (BNL) Model.

A. Uncertainty Estimation

There are two types of uncertainty that exist in model learning. The first type is model parameter uncertainty, also called epistemic uncertainty, represented with Σ in BNL. The second type is intrinsic uncertainty, also called aleatoric uncertainty, which originates from intrinsic randomness of the real model, represented with σ^2 in BNL. With the parameter uncertainty, the agent can be directed to explore and learn a better model more efficiently. While intrinsic uncertainty can be irrelevant or even damaging for exploration.

Instead of assuming a fixed noise variance σ^2, we model the joint distribution of $\mathbf{w}$ and σ^2: $p(\mathbf{w}, \sigma^2) = p(\mathbf{w}|\sigma^2)p(\sigma^2)$. We assume $\sigma^2 \sim \text{IG}(a_t, b_t)$, an Inverse Gamma distribution. The distribution of $\mathbf{w}$ conditioned on σ^2 is assumed to be $\mathbf{w}|\sigma^2 \sim \mathcal{N}(\mu_t, \sigma^2 \Sigma_t)$.

We set the prior hyper parameters: $\mu_0 = 0$, $\Lambda_0 = \lambda I$ and $a_0 = b_0 = \eta > 1$, then initially $\sigma_0^2 \sim \text{IG}(\eta, \eta)$ and $\mathbf{w}|\sigma_0^2 \sim \mathcal{N}(0, \sigma_0^2/\lambda I)$. After observing $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, the closed-form updates for the exact posterior inference [47] is:

$$\Sigma_t = (\Phi\Phi^T + \Lambda_0)^{-1}, \quad \mu_t = \Sigma_t(\Lambda_0\mu_0 + \Phi\Phi^T).$$

The parameters of the Inverse Gamma distribution of σ^2 can be updated with

$$a_t = a_0 + t/2$$

$$b_t = b_0 + \frac{1}{2}\left(\mathbf{y}^T\mathbf{y} + \mu_0^T\Sigma_0\mu_0 - \mu_t^T\Sigma_t^{-1}\mu_t\right).$$

The intrinsic uncertainty can be calculated by sampling from the Inverse Gamma distribution: $\sigma^2 \sim \text{IG}(a_t, b_t)$. And the output variance resulting from the parameter uncertainty is

$$\varsigma^2 = \phi_*^T \Sigma_t \phi_*. \tag{7}$$

B. Information Gain

The entropy of the BNL Model can be defined as the entropy of the posterior $\mathbf{w}$ (Eq. (3)):

$$\mathbb{H}(\mathbf{w}) = \frac{1}{2}\log|\Sigma_t| + \frac{N}{2}(\log 2\pi e).$$

After observing new data $\mathcal{D}_* = (\mathbf{x}_*, \mathbf{y}_*)$ the information gain about the model is the decrease in entropy:

$$\begin{aligned}\mathbb{I}(\mathbf{w}; \mathcal{D}_*) = \mathbb{H}(\mathbf{w}) - \mathbb{H}(\mathbf{w}; \mathcal{D}_*) &= \frac{1}{2}\log\left|\frac{\sigma^{-2}\phi_*^T\phi_* + \Sigma_t^{-1}}{\Sigma_t^{-1}}\right| \\ &= \frac{1}{2}\log\left|1 + \frac{\phi_*^T\Sigma_t\phi_*}{\sigma^2}\right| = \frac{1}{2}\log\left|1 + \frac{\varsigma^2}{\sigma^2}\right|,\end{aligned} \tag{8}$$

where $\mathbb{H}(\mathbf{w}; \mathcal{D}_*) = -\frac{1}{2}\log|\sigma^{-2}\phi_*^T\phi_* + \Sigma_t^{-1}| + \frac{N}{2}(\log 2\pi e)$.

C. Value Function Approximation

In this paper, we consider the sequential decision-making problem, where an agent infers an action a in state x, according to the predicted value Q.

In discrete action scenarios, we model the predicted value Q_i with BNL Model, where for each action a_i we share the neural network $\phi(x)$ but the Bayesian linear parameter $\mathbf{w}_i$ is independent. Different from linear models that directly regress values Q_i on x, we independently train a deep neural network to learn a representation $\phi(x)$, and then use a Bayesian linear regression to regress Q_i on $\phi(x)$. The network is only used to find good representation $\phi(x)$. In addition, we can update the network and the Bayesian linear regression at different time scales. In our experiments, the network is only updated after a number of points have been collected. Then, we perform a forward pass on all the training data to obtain $\phi(x)$, which is then fed to the Bayesian linear regression.

In continuous control problem, the value function $Q(x, a)$ is approximated with BNL Model which takes (x, a) as input and outputs the value estimation. Then the BNL Model extracts a representation $\phi(x, a)$ and regresses $Q(x, a)$ on $\phi(x, a)$.

The uncertainty of the value function can be calculated according to the BNL Model. The output variance resulting from the parameter uncertainty is calculated with Eq. (7), representing the epistemic uncertainty of the learned value function. The noise variance can be sampled from the updated Inverse Gamma distribution: $\sigma^2 \sim \text{IG}(a_t, b_t)$, representing the intrinsic randomness of the environment. Sequentially estimating the parameter uncertainty Σ_t allows the agent to explore accordingly and to improve its understanding of the environment adaptively.

V. Exploration With Bayesian Neural Linear Policy

With BNL Model approximating the value function, the uncertainty of the value function can be represented and can be used to direct the exploration.

Information-Directed Sampling (IDS) [23], [48] is a Multi-Armed Bandit algorithm. It chooses the action with the smallest regret-information ratio: $\frac{\hat{\Delta}(a)^2}{\mathbb{I}(a)}$, where $\mathbb{I}(a)$ is the information gain and regret $\hat{\Delta}(a)$ is the loss in reward for choosing action a. IDS trades-off between incurring small regret and acquiring more information at each step. In this section, we extend the result of IDS [23] to contextual bandit and MDP problems with discrete action space, taking advantage of the powerful representation and efficient uncertainty calculation in the BNL model. For continuous control problems, we sample a value function from the Bayesian linear model and propagate the gradient to the policy function to perform policy optimization.

A. Exploration in Contextual Bandit

We cannot directly compute the expected regret in contextual bandit since it depends on the unknown optimal action a^*. Instead, IDS uses a conservative regret estimate, which extended to the contextual bandit is

$$\hat{\Delta}_a(x) = \max_{a' \in \mathcal{A}} u_{a'}(x) - l_a(x), \tag{9}$$

where $[l_a(x), u_a(x)]$ is a confidence interval which contains the true expected action value $Q_a(x)$ with high probability. Based on the mean and variance estimate of the action value $Q_a(x)$, the confidence intervals can be defined as:

$$l_a(x) = \mu_a(x) - \lambda \varsigma_a(x), \quad u_a(x) = \mu_a(x) + \lambda \varsigma_a(x) \tag{10}$$

where $\varsigma^2 = \phi_*^T \Sigma_t \phi_*$ is derived in Section IV-A. The first term in the right of Eq. (9) corresponds to the maximum plausible action value could receive at a state x, while the last term $l_a(x)$ lower-bounds the value of the chosen action. As a result, Eq. (9) provides a conservative estimate of the true regret.

According to Eq. (8), the information gain for choosing action a and observing Q_a can be calculated with

$$\mathbb{I}_a(x) = \frac{1}{2} \log \left| 1 + \frac{\varsigma_a(x)^2}{\sigma_a(x)^2} \right|. \tag{11}$$

In particular, the information gain $\mathbb{I}_a(x)$ is small for actions with little epistemic value uncertainty or with high intrinsic randomness in observed value. $\sigma_a(x)^2$ is explicitly dependent on the selected action a and state x, which allows the exploration strategy to account for heteroscedastic noise.

With the regret and information gain calculated in Eq. (9) and Eq. (11), agent can choose the action with minimum regret-information ration: $\frac{\hat{\Delta}_a(x)}{\mathbb{I}_a(x)}$. Then, the information-directed exploration through the BNL model balances between incurring minor regret and acquiring more information at each step, i.e. exploration and exploitation trade-off.

B. Exploration in Discrete Action Space

Similar to BDQN [41], we use BNL model to approximate the action value function in MDP: $Q^\pi(x, a)$ (Eq. (4)), with independent Bayesian linear regression parameter $\mathbf{w}_a$ for the value of each discrete action a.

In order to apply the Information-Directed exploration in MDP, we have to define the regret $\Delta^\pi(x, a)$ and information gain $\mathbb{I}(x, a)$. It is straightforward to extend the instantaneous regret calculation in contextual bandit to MDP, i.e. Eq (9), with $[l_a(x), u_a(x)]$ representing the confidence interval of $Q^\pi(x, a)$. In order to calculate the confidence interval and the information gain $\mathbb{I}(x, a)$, we need to estimate both epistemic and intrinsic uncertainty of $Q^\pi(x, a)$, represented with $\varsigma_a(x)^2$ and $\sigma_a(x)^2$ respectively.

With the Bayesian linear parameter uncertainty in the BNL model, epistemic uncertainty can be calculated with $\varsigma_a(x)^2 = \phi_a(x)^T \Sigma_a \phi_a(x)$, where Σ_a is calculated according to Eq. (3). For estimating the intrinsic uncertainty we resort to the distributional reinforcement learning (C51) [43]. Instead of the expected action value, distributional reinforcement learning estimates a full distribution of the action value: $Z_\theta(x, a)$. However, according to the previous study [49], the estimated value distribution $Z_\theta(x, a)$ is a mixture of the parametric and intrinsic uncertainties. So we calculate the intrinsic uncertainty by removing the parametric uncertainty from $Z_\theta(x, a)$:

$$\sigma_a^2(x) = \frac{Var(Z_\theta(x, a))}{\frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} Var(Z(x, a'))} - \varsigma_a^2(x), \tag{12}$$

where we normalize the distribution variance $Var(Z_\theta(x, a))$, allowing the agent to account for the numerical differences across environments as in [40].

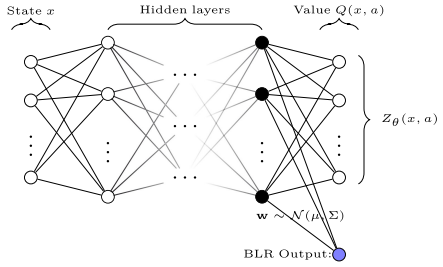


Fig. 1. BNL model structure for the action value approximation in MDP.

In MDP, the intrinsic uncertainty of action value calculated with Eq. (12) has advantages over the Gaussian noise estimation with inverse gamma distribution: $\sigma^2 \sim \text{IG}(a_t, b_t)$, because such intrinsic uncertainty can be propagated through time with distributional reinforcement learning, thus providing a global uncertainty estimation.

To perform the distributional reinforcement learning in the BNL model, we share the hidden layers with the value distribution approximation model. The BNL model regresses between the state x and the mean of the target output distribution $r_t + \gamma Z_\theta(x_{t+1}, a')$ (in Eq. (5)), to ignore the intrinsic noise of the action value. The structure of the combined value function model is illustrated in Fig. 1. It only shows the value output of one action, including the value distribution and the output of Bayesian linear regression. The hidden layers are shared between the value models of each discrete action.

Then, the information gain and regret-information ratio can be calculated as in contextual bandit. The agent takes action with minimum regret-information ratio $\frac{\hat{\Delta}^\pi(x,a)}{\mathbb{I}(x,a)}$ to achieve the information-directed exploration.

C. Exploration in Continuous Action Space

Based on DDPG [45], we use BNL model to approximate the action value function: $Q^{\pi_\alpha}(x,a)$, where $a = \pi_\alpha(x)$. In DDPG, policy parameter is optimized by maximizing $Q^{\pi_\alpha}(x,a)$ with regard to α and the trade-off between exploration and exploitation is performed by annealing the additive action noise. While we perform exploration by sampling the linear parameter from its posterior $\mathbf{w} \sim \mathcal{N}(\mu_t, \Sigma_t)$ before each update and constructing a specific action value function which is maximized by updating the policy.

Policy gradients computed by maximizing the sampled action value function may lead to better exploration directly in the policy space. And it bears some similarity with Thompson Sampling [20] where the action value function guides the exploration in the action space.

VI. Adaptive Formation With Neural Linear Policy

A. Problem Formulation

We consider a multi-robots scenario where n robots navigate to their own goal positions while keeping in formation and avoiding m other moving robots (dynamic obstacle). We formulate the multi-robot adaptive formation problem as a sequential decision-making problem, where a distributed collision avoidance policy is built on top of the consensus-based formation controller.

Each robot's state is composed of the robot's observable states and unobservable states, $s_{i,t} = [s^o_{i,t}, s^h_{i,t}]$. The observable states are the robot's position, velocity and radius, $s^o_{i,t} = [p_x, p_y, v_x, v_y, r] \in \mathbb{R}^5$. The unobservable states are the robot's goal position, maximum speed, and orientation, $s^h_{i,t} = [p_{gx}, p_{gy}, v_{max}, \psi] \in \mathbb{R}^4$. The action is a speed and change of heading angle $u_t = [v_t, \delta\psi_t] \in \mathbb{R}^2$. The observable states of all $n+m-1$ other robots are concatenated into $\tilde{\mathbf{S}}^o_{i,t}$.

B. Consensus-Based Formation Planning of Multi-Robots

According to the protocol of consensus-based formation control Eq. (1), the consensus-based formation velocity for each robot i is

$$\mathbf{v}_i(t) = \alpha \sum_{j=1}^{N} a_{ij} \left[\mathbf{v}_j(t) - \gamma \left(\mathbf{p}_i(t) - \mathbf{p}_j(t) - \mathbf{d}_{ij}\right)\right] + \beta \left[\mathbf{p}_{i,g} - \mathbf{p}_i(t)\right] \tag{13}$$

where $\mathbf{d}_{ij} := \mathbf{p}_{id} - \mathbf{p}_{jd}$ is the desired displacement between the robot i and the robot j. $\mathbf{p} = [p_x, p_y], \mathbf{v} = [v_x, v_y]$ are the vectors of the position and velocity corresponding to $\dot{x}, x$ in Eq. (1) respectively. $\mathbf{p}_{i,g}$ is the goal position of robot i. To drive each robot move to its goal, we add $beta[\mathbf{p}_{i,g} - \mathbf{p}_i(t)]$ to Eq. (1, where α and β are two weighting coefficients.

C. Extending Formation Planning to Dynamic Obstacle Avoidance With RL

Based on the consensus-based formation velocity $\mathbf{v}_i(t)$ in Cartesian coordinates, the speed and change of heading angle $u_t = [v_t, \delta\psi_t]$ can be constructed. To avoid collision with other moving robots, a modification of the formation velocity is constructed as a policy: $\pi : (s_t, \tilde{\mathbf{S}}^o_t) \to a_t = [\delta v_t, \delta\psi_t] \in \mathbb{R}^2$. The policy should be optimized with the objective of minimizing expected time to goal $\mathbb{E}[t_g]$ while avoiding collision with other robots,

$$\begin{aligned} &\arg\min_{\pi_i} \mathbb{E}\left[t_g \mid \mathbf{s}_i, \tilde{\mathbf{S}}^o_i, \pi_i\right] \\ &\text{s.t. } \left\|\mathbf{p}_{i,t} - \tilde{\mathbf{p}}_{j,t}\right\|_2 \geq r_i + r_j \quad \forall j \neq i, \forall t \end{aligned}$$

where the expectation is with respect to the other agent's unobservable states (intents) and policies. The optimization problem is subject to the collision avoidance constraint and is solved by optimizing policy π_i with reinforcement learning.

By executing the combined action:

$$\begin{aligned} &\pi_i\left(\mathbf{s}_{i,t-1}, \tilde{\mathbf{S}}^o_{i,t-1}\right) + \mathbf{v}_i(t-1) \\ &= a_i(t-1) + u_i(t-1) = [\delta v_t, \delta\psi_t] + [v_t, \delta\psi_t] \end{aligned} \tag{14}$$

the agents' kinematics are updated:

$$\mathbf{p}_{i,t} = \mathbf{p}_{i,t-1} + \Delta t \cdot (a_i(t-1) + u_i(t-1))\, \forall i \tag{15}$$

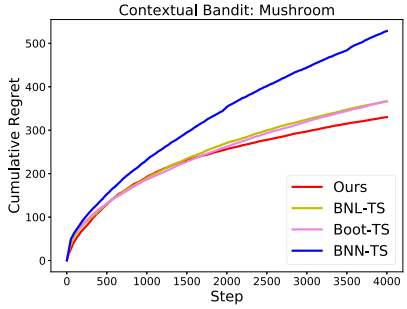


Fig. 2. Average cumulative regrets in 20 independent experiments, plotted over the samples (steps) needed, smaller is better.

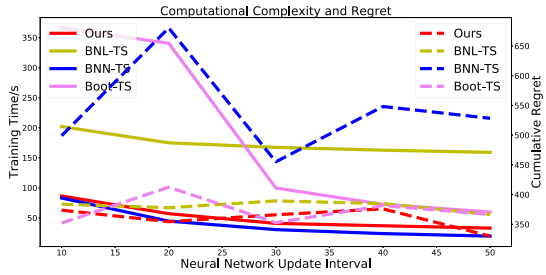


Fig. 3. Average overall training time (left axis) and the average final cumulative regrets (right axis), regarding update intervals.

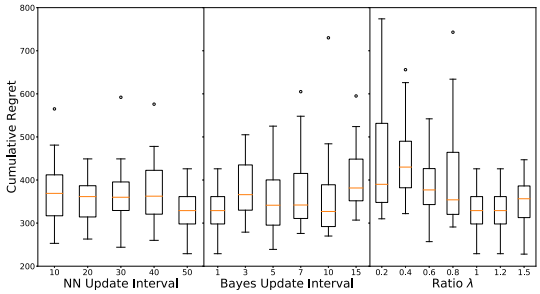


Fig. 4. A boxplot of the average final cumulative regrets achieved by our method, plotted over the changes in three hyper parameters.

D. Information-Directed Exploration to Train the Policy

We construct the obstacle avoidance policy π_i with our Neural Linear Policy, where the joint world state, $\mathbf{s}^{join} = (\mathbf{s}_{i,t}, \tilde{\mathbf{S}}^o_{i,t})$, is as input to the Neural Linear Model. The linear and angular speed is chosen as the action of each robot, which is discretized to form action space. With the BNL Model, the value of discrete action of obstacle avoidance can be evaluated with uncertainty which directed the exploration of robots, i.e. the data sample to train the model. A reward function is specified to award each robot for reaching its goal, and penalize each robot for getting too close or colliding with other robots.

$$R\left(\mathbf{s}^{join}, \mathbf{a}\right) = \begin{cases} 1 & \text{if } \mathbf{p} = \mathbf{p}_g \\ -0.25 & \text{if } \left\|\mathbf{p}_{i,t} - \tilde{\mathbf{p}}_{j,t}\right\|_2 \geq r_i + r_j \quad \forall j \neq i \\ -0.01 & \text{otherwise} \end{cases} \tag{16}$$

In sampling data with Information-Directed Exploration, we terminate one episode when there is a collision or all robots arrive at their goals. So, the reward -0.01 promotes the robot to move to the goal as soon as possible. With the uncertainty directing the exploration (data sampling) and reward function directing the optimization, the distributed obstacle avoidance

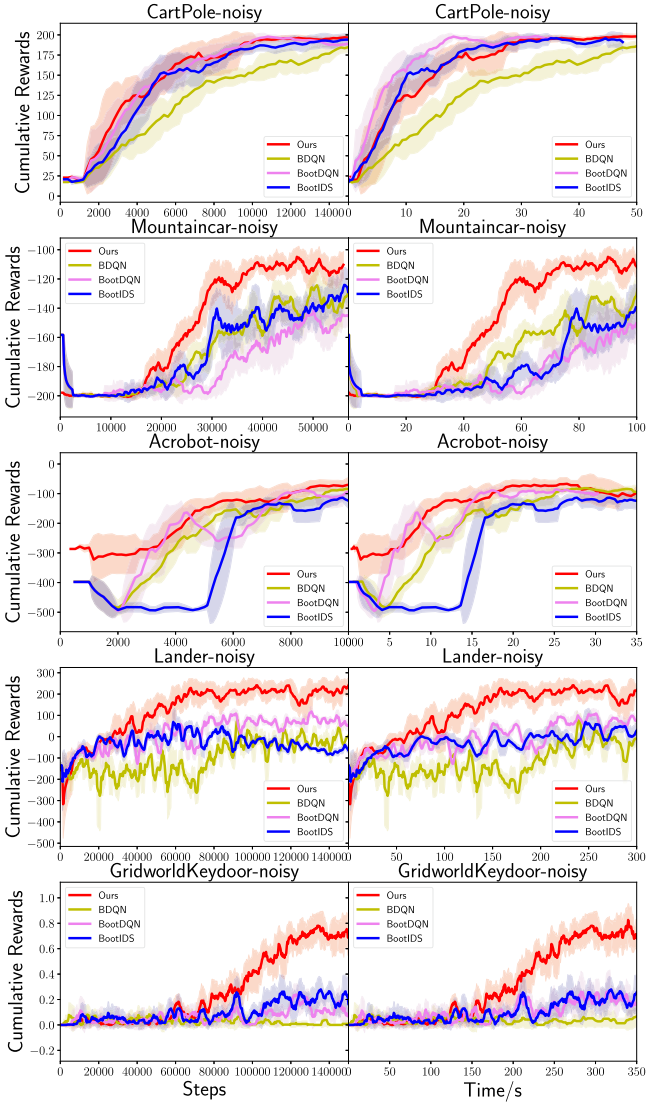


Fig. 5. Comparisons in MDP. The solid lines and shades represent te mean and standard deviation in five independent experiments.

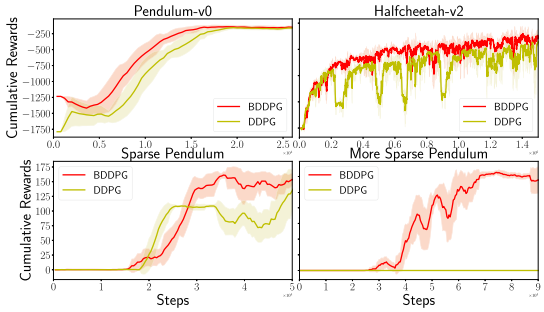


Fig. 6. Experiments in Continuous Control Tasks. The solid lines and shades represent the mean and standard deviation in five independent experiments.

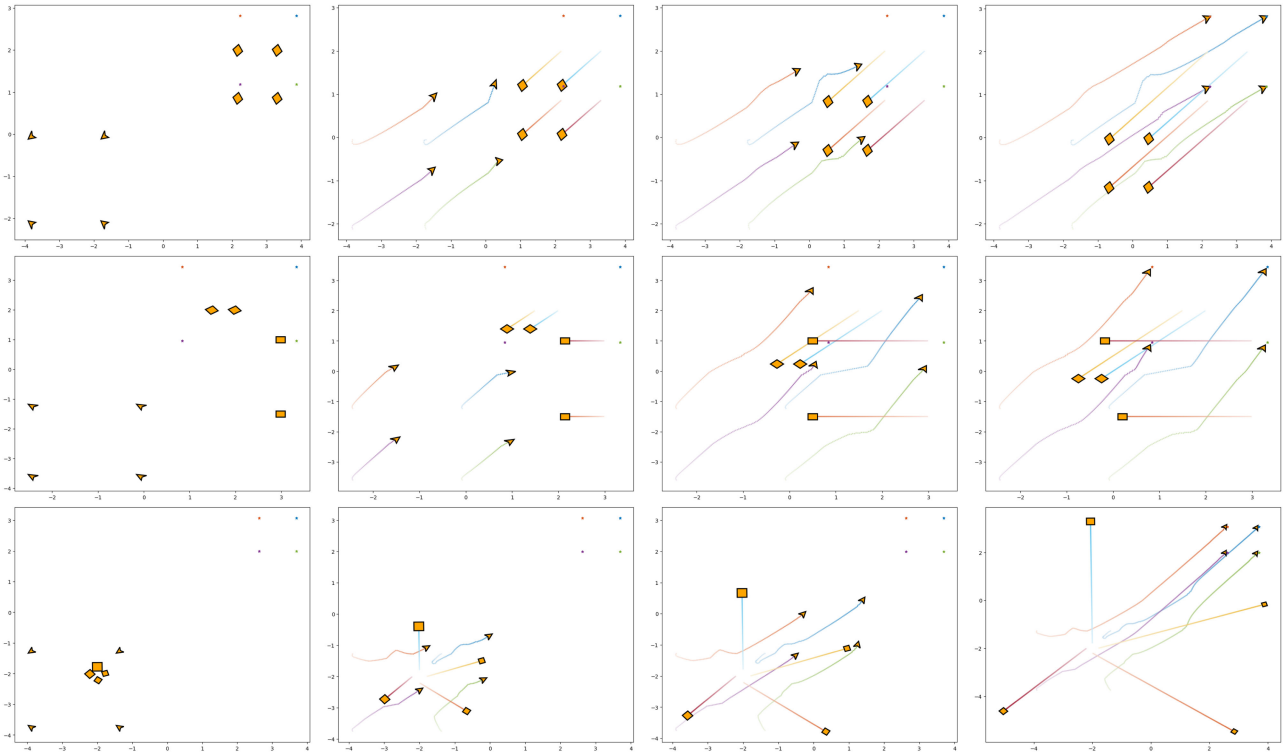


Fig. 7. Formation movement with dynamic obstacles moving in opposite direction (top), moving in side direction (middle) and spreading out(bottom).

policy can be optimized efficiently and augment the formation velocity to achieve adaptive formation.

VII. Experiments

A. Contextual Bandit

We test our method in the classic Mushroom dataset [50], which contains 22 attributes per mushroom and two classes: poisonous and safe. We define a bandit problem where a random mushroom is selected from the dataset, and the agent decides to eat or not according to the mushroom's attribute. If the agent does not eat the provided mushroom, the reward is 0. Eating a safe mushroom provides a reward that follows the Gaussian distribution $r \sim \mathcal{N}(1, 1)$, while eating a poisonous mushroom delivers a reward $r \sim \mathcal{N}(-3, 1)$. The attributes do not indicate poisonous or safe directly, but the relationship should be learned from the rewards. The goal of the agent is to minimize the cumulative regret in n decisions. In our experiments, we set $n = 4000$.

Thompson Sampling is a popular exploration strategy and often provides better empirical results than UCB [37], so we choose Thompson Sampling as the baseline, using the Bayesian neural network [51], Bootstrapped neural networks [15] and the BNL model to approximate the posterior. We abbreviate these algorithms as "BNN-TS," "Boot-TS" and "BNL-TS" respectively. In our experiments, each neural network has only one hidden layer with 50 units. The BNL model builds the Bayesian linear regression, instead of the full-connected layer, on the top of the hidden units. The training batch size is 512 in all experiments. The neural network update interval is 50 steps for all algorithms, and we update the Bayesian linear parameters $\mathbf{w}$ in every step for the BNL model.

We compare our method with others in the cumulative regret. Fig. 2 shows that our method incurs the smallest cumulative regrets over the whole sequential decision-making process, demonstrating the sample efficiency of our information-directed exploration.

Neural network training takes up the central part of computing resources. The overall training time varies a lot by changing the hyperparameter: network update interval. We compare the final time consuming and the cumulative regrets incurred by these methods with different neural network update intervals. All experiments are conducted on the same platform (Intel i7-6700 CPU and TITAN X GPU).

Though all the training time decreases as the update interval increases, our method takes less time to finish the whole process, as shown in Fig. 3. The Bootstrapped model takes more time for training, because there are several neural networks trained at the same time. "BNN-TS" takes the least training time, where the decision is the direct output of the neural network. However, "BNN-TS" incurs terrible cumulative regret due to the unstable training of the Bayesian neural network, here we minimize the evidence lower bound following [51]. From the performance of "BNL-TS," we can see that exploration with Thompson Sampling, which need sampling from the parameter posterior at every step, is also time-consuming. In contrast, our method calculates the regret-information ratio and explores according to the ratio without sampling. Thus, as shown in Fig. 3 our method is the only one that incurs the least cumulative regret and achieves efficiency in computation at the same time.

Apart from the neural network update interval, our method has two important hyperparameters: the Bayesian parameter update interval and λ in confidence interval calculation, Eq. (10). Fig. 4 demonstrates that the performance of our method is not very sensitive to these hyperparameters. But it is still obvious that the performance deteriorates as the Bayesian parameter is updated less frequently. Too small λ reduces the confidence interval and biases the estimation of the expected regret. As a result, it incurs a huge regret as shown in the last subfigure in Fig. 4.

B. Markov Decision Process

We evaluate our method in MDP with five simulated tasks: "CartPole-v0," "Acrobot-v0," "MountainCar-v0," "LunarLander-v2" and "Gridworld-Door & Key". The first four environments are provided in OpenAI Gym. "Gridworld-Door & Key" is from Minimalistic Gridworld [52], where the agent must pick up a key to unlock a door then arrive at a prescribed goal position across the door. A reward is received only after the task is accomplished. We modify these original tasks by adding heteroscedastic noises to all the environments' original rewards, with variance be 0.1 times of the original rewards.

We compare our method with Bootstrapped DQN [15], Bayesian DQN [41], and Bootstrapped IDS [40], abbreviated as "BootDQN," "BDQN" and "BootIDS". In "BootDQN" and "BootIDS," the value function is approximated with multiple neural networks in ensemble. "BDQN" uses Bayesian neural liner model but with Thompson Sampling for exploration as that in "BootDQN". "BootIDS" explores with information-directed sampling and estimates the uncertainty with the variance of multiple neural networks' outputs.

In our experiments, all algorithms use the same full connected neural network with two hidden layers and 64 units for each. The learning rates decrease from 0.001 to 0.00001 during all the training processes, and the batch size is 32. All the experiments below use the same platform with Intel i7-8700 CPU and GTX1080Ti GPU. In Fig 5, we present the experiments' results from the perspective of sample and computation time needed for training.

Our method achieves competitive results in all five tasks, both in sample efficiency and in computation (time) efficiency. Especially in the tasks with sparse rewards, "LunarLander-v2" and "Gridworld-Door & Key," our method outperforms other methods by a large margin. Though "BDQN" models the action value function with BNLmodel, it could not accomplish the last two tasks which are difficult with sparse rewards. We think that the noisy rewards result in an action value with heteroscedastic noise, which is challenging for "BDQN" and "BootDQN". In comparison with "BootIDS" which also uses the IDS strategy for exploration, our method has better performance, taking advantage of the efficiency of the BNL model in uncertainty estimation.

C. Continuous Control Tasks

In continuous control tasks, we build the value function with the BNL model and optimize the policy with policy gradient method. We call the method Bayesian DDPG (BDDPG). To demonstrate the performance of BDDPG, we compare it with the original DDPG in two continuous control tasks from OpenAI Gym: "Pendulum-v0" and "Halfcheetah-v2". We also modify "Pendulum-v0" to be with sparse rewards, where at each time step "Sparse Pendulum" gets a reward 1 for near upright within 0.01 (rad) and "More Sparse Pendulum" gets a reward 1 for near upright within 0.001 (rad).

Fig 6 presents the results in continuous control tasks and demonstrates that our improved exploration with BNLmodel does not enjoy many advantages in the continuous control tasks with dense reward feedback. However, in tasks with more sparse reward feedback, efficient exploration with Bayesian model plays a more critical role.

D. Adaptive Formation

Through extensive experimental analysis using randomly generated point-to-point formation tasks, we further validate our BNL policy in the dynamic obstacle avoidance of multiple robots in formation. The BNL model is constructed to evaluate the state-action as in (V-B). The actions are constructed with the combination of discrete speeds and heading angle changes $u_t = [v_t, \delta\psi_t]$. For each robot in formation, an independent BNL model is built with observation vector $\tilde{\mathbf{S}}^o_{i,t}$ as input which concatenates the robot's full states and all $n+m-1$ other robots' observable states. We treat the action generated with the BNL policy as a modification to the formation velocity Eq. (13). To calculate the formation velocity, the communication topology among the robots in formation can be any directed graph, including a spanning tree.

In the experiments, BNL model is constructed with 64 nodes in the hidden layers. We set the maximum of time steps in one episode as 500, every time step represents 0.2 s, and the training batch size is chosen as 512. The learning rate decays from 0.001 to 0.00001 during the training process. Fig. (7) in our experiments show the trajectories of all robots. In those experiments, there are eight robots (marked from 0˜7) with unicycle dynamic and maximum turning angle in 0.2 seconds is set as 0.2π rad. The safe radius of every robot is 0.1 to avoid collision. Robot 0˜3 should be in formation, moving to their individual goal and avoiding collision with other robots. Robot 4˜7 are treated as dynamic obstacles which moving straightforward with different starts and goals. In our experiments, the dynamic obstacles are moving independently with different patterns, presented in Fig. (7) for example, in the opposite direction (top) and side direction to the robots in formation (middle). In the bottom of Fig. (7), dynamic obstacles are spreading out from the vicinity of original point. The experiments show that the policy trained with our method drives multiple robots to goal positions and avoids collision with other moving robots successfully while maintaining formation as much as possible.

VIII. Conclusion and Discussion

In this paper, we first focus on improving the efficiency of exploration w.r.t. the sample and computation in sequential decision making. We approximate the state-action value with the BNL model that combines Bayesian linear regression with

neural network and achieves computational efficiency in uncertainty representation. By defining the information gain about the learned BNL model and the regret for each decision step, we extend IDS to the contextual bandit problem and MDP. In MDP, we estimate the global intrinsic uncertainty of the action value with distributional reinforcement learning, considering a mixture of the aleatoric and epistemic uncertainty in it. We perform Thompson sampling in the BNL value function, based on which a policy can be optimized to achieve efficient exploration in continuous control tasks. Several experiments demonstrate the improved efficiency of our proposed method in the Contextual Bandit problem and MDP. We find that the BNL model provides an efficient estimation of uncertainty, based on which information gain and regret can be calculated efficiently. Thus, our information-directed exploration strategy provides an efficient solution for sequential decision making.

We then apply the BNL model to Multi-Robot learning dynamic obstacle avoidance in formation movement by training the BNL policy with information-directed exploration. Central to our method's scalability and success combines a consensus-based formation movement and an on-demand collision avoidance strategy. The strategy is focusing on obstacle avoidance and is trained with reinforcement learning. Our method is validated through many simulations where dynamic obstacles move with different patterns relative to the robots.

Thus, our information-directed exploration strategy provides an efficient solution for sequential decision making. However, our method has limits in representing the uncertainty of very deep neural networks. Further research can be conducted to perform our method on the representations that are learned with unsupervised learning such as variational auto-encoder, to further improve the efficiency and generalization. The configuration of multi-robots is sensitive such as the number of robots and the safe radius. Randomizing such configuration parameters to train a robust policy deserves further research.

REFERENCES

[1] J. Borenstein and Y. Koren, "The vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.

[2] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[3] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Automat. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[4] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, 2002.

[5] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Edmonton, Canada, 2005, pp. 2210–2215.

[6] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 5954–5961.

[7] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 477–483.

[8] F. C. Yu, L. Miao, M. F. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 285–292.

[9] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.

[10] P. Long, W. Liu, and P. Jia, "Deep-learned collision avoidance policy for distributed multi-agent navigation," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 656–663, Apr. 2017.

[11] S. Qi and S. C. Zhu, "Intent-aware multi-agent reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7533–7540.

[12] M. Li, R. Jiang, S. S. Ge, and T. H. Lee, "Role playing learning for socially concomitant mobile robot navigation," *CAAI Trans. Intell. Technol.*, vol. 3, no. 1, pp. 49–58, 2018.

[13] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.

[14] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[15] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," *Advances in Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4026–4034.

[16] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4072–4081.

[17] M. J. Strens, "A Bayesian framework for reinforcement learning," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 943–950.

[18] I. Osband, D. Russo, and B. Van Roy, "(More) efficient reinforcement learning via posterior sampling," in *Proc. Neural Inf. Process. Syst.*, 2013, pp. 3003–3011.

[19] I. Osband, B. V. Roy, D. J. Russo, and Z. Wen, "Deep exploration via randomized value functions," *J. Mach. Learn. Res.*, vol. 20, no. 124, pp. 1–62, 2019.

[20] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[21] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, 2002.

[22] I. Osband, B. V. Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2377–2386.

[23] J. Kirschner and A. Krause, "Information directed sampling and bandits with heteroscedastic noise," in *Proc. Conf. Learn. Theory*, 2018, pp. 358–384.

[24] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," *Comput. Ence Dept.*, vol. 98, pp. 1–4, Oct. 1998.

[25] A. Stentz, "The focussed D* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659, Aug. 1995.

[26] M. Čáp *et al.*, "Multi-agent RRT: sampling-based cooperative pathfinding," in *Proc. 2013 Int. Conf. Auton. Agents Multi-Agent Syst.*, May 2013, pp. 1263–1264.

[27] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Auton. Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[28] T. Schouwenaars *et al.*, "Mixed integer programming for multi-vehicle path planning," in *Proc. 2001 Eur. Control Conf. (ECC)*, IEEE, Sep. 2001, pp. 2603–2608.

[29] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robot. Res.*, Springer, Berlin: Heidelberg, 2011, pp. 3–19.

[30] J. Snape, J. v. d. Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.

[31] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1527–1533.

[32] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 31–36.

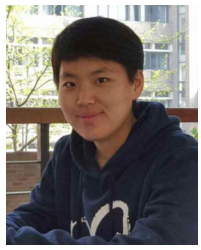
[33] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[34] M. Plappert *et al.*, "Parameter space noise for exploration," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–18.

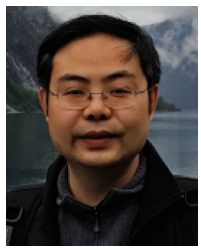
[35] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn.*, vol. 2017, 2017, pp. 2778–2787.

[36] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "UCB exploration via Q-ensembles," 2017, *arXiv:1706.01502*.
[37] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2249–2257.
[38] T. M. Moerland, J. Broekens, and C. M. Jonker, "Efficient exploration with double uncertain value networks," *Deep Reinforcement Learn. Symp., NIPS*, 2017.
[39] D. Mackay, "Information-based objective functions for active data selection," *Neural Comput.*, vol. 4, no. 4, pp. 590–604, 1992.
[40] N. Nikolov, J. Kirschner, F. Berkenkamp, and A. Krause, "Information-directed exploration for deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–16.
[41] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar, "Efficient exploration through bayesian deep Q-networks," in *Proc. Inf. Theory Appl. Workshop*, 2018, pp. 1–9.
[42] R. Dearden, "Bayesian Q-learning," in *Proc. Nat. Conf. Artif. Intell.*, 1998, pp. 761–768.
[43] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 449–458.
[44] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[45] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *ICLR*, pp. 1–14, Jan. 2016.
[46] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[47] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[48] D. Russo and B. Van Roy, "Learning to optimize via information-directed sampling," in *Proc. Neural Inf. Process. Syst.*, 2014, pp. 1583–1591.
[49] B. Mavrin, H. Yao, L. Kong, K. Wu, and Y. Yu, "Distributional reinforcement learning for efficient exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4424–4434.
[50] J. Schlimmer, "Mushroom records drawn from the audubon society field guide to north American mushrooms," *GH Lincoff*, New York, NY, USA, 1981.
[51] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1861–1869.
[52] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for OpenAI gym," 2018. [Online]. Available: https://github.com/maximecb/gym-minigrid

Junjie Cao received the B.S. degree in mechanical engineering and automation from Nanjing Tech University, Nanjing, China, in 2014, and the M.S. degree in 2017 in mechatronics from Zhejiang University, Hangzhou, China, where he is currently working toward the Ph.D. degree with the College of Control Science and Engineering. His current research interests include machine learning, sequential decision making, and robotics.



Yujie Wang received the B.S. degree in 2018 from Zhejiang University, Hangzhou, China, where she is currently working toward the Ph.D. degree with the Academy of Humanities and Social Sciences. Her main research interests include MIDAS data mining, thick-tail heterogeneous economic research. She also committed to applying advanced deep learning techniques in financial prediction, reinforcement learning methods in Forex scenario.



Yong Liu (Member, IEEE) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively. He is currently a Professor with the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University. He has authored or coauthored more than 60 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, and information fusion.

Xuesong Ni received the B.S. degree in automation from Northwestern Polytechnical University, Xi'an, China, in 2010 and the M.S. degree from Beihang University, Beijing, China in 2013. His current research interests include robotics and information fusion.